(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) Übersetzung der europäischen Patentschrift

(87) EP 0 760 119 B1

(10) DE 695 04 192 T 2

(51) Int. Cl.[6]: G 06 F 7/52

(21) Deutsches Aktenzeichen: 695 04 192.4
(86) PCT-Aktenzeichen: PCT/FR95/00655
(86) Europäisches Aktenzeichen: 95 920 958.6
(87) PCT-Veröffentlichungs-Nr.: WO 95/32465
(86) PCT-Anmeldetag: 18. 5. 95
(87) Veröffentlichungstag der PCT-Anmeldung: 30. 11. 95
(87) Erstveröffentlichung durch das EPA: 5. 3. 97
(87) Veröffentlichungstag der Patenterteilung beim EPA: 19. 8. 98
(47) Veröffentlichungstag im Patentblatt: 24. 12. 98

(30) Unionspriorität:
9406214 20. 05. 94 FR

(73) Patentinhaber:
SGS-Thomson Microelectronics S.A., Gentilly, FR

(74) Vertreter:
Beetz und Kollegen, 80538 München

(84) Benannte Vertragstaaten:
DE, FR, GB, IT

(72) Erfinder:
CURTET, Joel, F-38600 Fontaine, FR

(54) ANORDNUNG ZUR DIGITALEN DURCHFÜHRUNG EINER DIVISIONSOPERATION

DE 695 04 192 T 2

DE 695 04 192 T 2

Die Erfindung betrifft eine Vorrichtung zur digitalen Durchführung einer Divisionsoperation, die dazu geeignet ist, Zwischenreste nicht zu berücksichtigen. Sie findet insbesondere Anwendung bei Berechnungen von binären Divisionen von ganzen Zahlen oder Brüchen als Festkommazahlen. Sie findet insbesondere bei Signalverarbeitungsprozessoren Anwendung.

Die binäre Divisionsberechnung verwendet hauptsächlich Vergleichs- und Subtraktionsoperationen. Es müssen einige Probleme wie das Format der Operanden (bruchzahlig oder ganzzahlig), das Vorzeichen des Ergebnisses, das Auftreten von ungültigen Teilergebnissen berücksichtigt werden, was ein ungültiges Ergebnis nach sich ziehen kann etc. Im allgemeinen wird für die Division ein Register verwendet, das Akkumulator genannt wird, das anschließend für jeden Schritt der Division den berechneten Rest bei den am höchsten gewichteten Bits und den Quotient bei den am niedrigsten gewichteten Bits und ein Quellenregister für den Nenner enthält.

Für die unabhängige Verarbeitung von Brüchen und ganzen Zahlen, mit oder ohne Vorzeichen, wird eine Lösung verwendet, die Nichtberücksichtigung der Zwischenreste genannt wird und auf dem Vorzeichen des Ergebnisses der Subtraktion oder der Addition des Zwischenrestes und des Nenners beruht. Wie es der Titel andeutet, kann es sein, daß die Zwischenergebnisse nicht richtig sind: bei einem Schritt einer gegebenen Division müßte, wenn das im vorherigen Schritt berechnete Bit des Quotienten Null ist, da man dennoch den Nenner abgezogen haben wird, der Nenner zum Zwischenrest

addiert werden, um den richtigen Zwischenrest wiederzuerhalten.

Für jeden Divisionsschritt muß der Quotient und der neue Zwischenrest berechnet werden.

Mehrere Ausführungen des Algorithmus sind in einer Rechenvorrichtung möglich. Erfindungsgemäß ist es von Interesse bei einer Durchführung das gleiche Register, das zu Beginn den Dividenden enthält, zu verwenden, um nachfolgend die Zwischenreste und den Quotienten während der Division zu speichern. Der Rest wird bei den am höchsten gewichteten Bits und der Quotient in den am niedrigsten gewichteten Bits gespeichert. Diese Durchführung weist folgende aufeinanderfolgenden Operationen bei jedem Divisionsschritt i auf:

- Verschieben des Registers mit dem Zwischenrest und dem Quotienten um eine Position nach links;

- Einführung der Invertierung des Bits des komplementären Quotienten, der beim vorherigen Schritt i-1 in der Position am weitesten rechts im Register berechnet wurde;

- Berechnung des Zwischenrestes durch Addition oder Subtraktion des auf den hoch gewichteten Bits des Registers angeordneten Nenners mit dem binären Wert des komplementären Quotienten, der beim vorherigen Schritt i-1 der Division berechnet wurde;

- Berechnung des Bits des komplementären Quotienten für den folgenden Schritt i+1 in Abhängigkeit vom Vorzeichen des neuen Zwischenrests und vom Vorzeichen des Nenners.

Dieser Vorgang setzt sich fort, bis n Verschiebungen unter der Berücksichtigung des Nenners mit einer Größe von n Bits durchgeführt worden sind und bis ein Quotient einer Größe von n Bits erzeugt wurde.

Aber dieser Algorithmus setzt voraus, daß eine gewisse Anzahl von Voraussetzungen überprüft worden sind, damit der endgültige Quotient richtig ist.

Wenn insbesondere während eines Divisionsschrittes ein Zwischenrest Null auftritt, kann der nach dem letzten Schritt der Division erhaltene Quotient falsch sein. Die Zahl Null wird als positiv betrachtet. Für eine binäre Division wird das Vorzeichen des Restes eines Divisionsschrittes i mit dem Vorzeichen des Nenners verglichen. Wenn der Nenner negativ ist, wird der Rest Null als positiv angesehen und falsch interpretiert, was zu einer Addition des Nenners und nicht zu einer Subtraktion im folgenden Schritt der Division führt. Der endgültige Quotient ist also falsch. Dies wird korrigiert, indem eine "1" zu der Position mit niedrigstem Gewicht addiert wird.

Weiterhin ist das Erkennen eines Zwischenrests Null nicht einfach, da der Rest und der Quotient eine beachtliche Größe haben: bei jedem Divisionsschritt verliert der Rest ein Bit, während der Quotient in der Größe eines gewinnt: es sind daher Operanden von beachtlicher Größe in Abhängigkeit vom Divisionsschritt. Da sie in dem gleichen Register liegen, bräuchte man eine aufwendige Schaltung, um der Entwicklung der Größe des Restes in Abhängigkeit von dem Divisionsschritt zu folgen und zu bestimmen, bei wieviel Bits die Decodierung von Null erfolgen muß.

Als weitere Voraussetzung muß der Absolutwert des Zählers kleiner als der Nenner sein, sonst ist die Division nicht

möglich. Bei einer Division mit Vorzeichen muß man auch wissen, ob der Nenner und der Zähler das gleiche oder ein entgegengesetztes Vorzeichen haben, um auf der Position des Quotienten mit dem geringsten Gewicht eine Einheit zu addieren und so den Quotienten zu korrigieren oder nicht, wenn der Nenner und der Zähler ein entgegengesetztes Vorzeichen haben.

Es müssen also unterschiedliche Voraussetzungen berücksichtigt werden, um das richtige Resultat zu erhalten. Dies wird mit Hilfe von Interrupt-Routinen möglich, die vom Anwender je nach Anforderung der Anwendung geschrieben werden (Format, Größe der zu verarbeitenden Zahlen). Außer, daß diese Routinen zeitaufwendig sind, ist es auch möglich, daß sie nicht in allen Fällen geeignet sind, da sie der Division nicht schrittweise folgen: sie werden vorher ausgeführt, um zu prüfen, ob die Division möglich ist, oder nachher, um eventuell den erhaltenen Quotienten zu korrigieren.

Um die Durchführung des Algorithmus insbesondere das Problem der Zwischenreste Null zu vereinfachen, haben einige Konstrukteure dieser Rechenvorrichtungen das Format der Operanden am Eingang begrenzt. Einige Vorrichtungen arbeiten beispielsweise nur mit positiven Operanden oder nur mit positiven Nennern: das Problem der negativen Nenner existiert nicht mehr. Aber es werden andere Routinen benötigt, um die Vorzeichen zu testen und sie zu speichern und um die negativen Operanden in positive Operanden vor der Berechnung umzuwandeln, und eine Routine, um dem Quotienten nach der Berechnung das richtige Vorzeichen wiederzugeben.

Weiterhin ist ein Register für den Rest und ein Register für den Quotienten vorgesehen, um das Erkennen des Zwischenrestes Null zu erleichtern, wobei diese Lösung jedoch

die Steuerung dieser Register für die Division komplizierter macht.

Die Wahl der Register, der Ausnahmeroutinen, das Bringen in das richtige Format und das Korrigieren des Quotienten erhöht die Anzahl der Befehlszyklen und somit die Rechenzeit. Wenn man einen endgültigen korrekten Rest haben will, muß man noch unterschiedliche Bedingungen berücksichtigen, um festzustellen, ob er richtig ist, und um ihn zu korrigieren, wenn nicht.

Eine Aufgabe der Erfindung ist es, eine Rechenvorrichtung zu schaffen, die Schaltungen aufweist, um den Rest Null im Laufe der Division zu erkennen. Die Erfindung ist in Anspruch 1 definiert.

Wie beansprucht betrifft die Erfindung eine Vorrichtung zur digitalen Durchführung einer Divisionsoperation mit der Methode, den Zwischenrest nicht zu berücksichtigen, die eine arithmetische und logische Einheit aufweist, um den Zwischenrest zu berechnen, dadurch gekennzeichnet, daß sie einen Schaltkreis zum Erkennen eines Zwischenrestes Null im Laufe der Division aufweist.

Erfindungsgemäß weist der Nenner ein Format mit n Bits auf, der Zwischenrest hat ein Format mit p Bits, wobei $p \geq n$ gilt, die bündig auf den Bits mit dem höchsten Gewicht eines Akkumulationsregisters liegen. Der Test des Zwischenrestes beim Divisionsschritt i wird nach der Addition oder der Subtraktion des auf den n Bits mit dem höchsten Gewicht des Restes angeordneten Nenners zum Inhalt des Akkumulators durchgeführt und besteht darin zu testen, ob die Bits des neuen Restes mit dem höchsten Gewicht von der Position $(p - n)$ bis $(p - 1)$ zusammengezählt mit dem Bit mit niedrigem Gewicht nach der Position $(p - 1 - n)$ alle Null sind,

um ein Informationsbit auf 1, das anzeigt, daß der Rest teilweise Null ist, und auf Null zu setzen, wenn nicht, und, wenn dieses Informationsbit schon auf 1 gesetzt ist, nur das Bit auf der Position (p - 1 - n) zu testen, um das Informationsbit auf 1 zu halten, wenn es Null ist, oder es im gegenteiligen Fall auf Null zurückzusetzen.

Der Ausdruck Rest, der teilweise Null ist, bedeutet, daß man nur weiß, daß die höherwertigen Bits des berechneten Zwischenrests im Divisionsschritt i Null sind, wobei es der Test des Bits an der Position (p - 1 - n) bei den folgenden Divisionsschritten ermöglicht, nach und nach festzustellen, daß der Inhalt des Akkumulationsregisters nach links verschoben ist, wenn die anderen Bits des im Schritt i gefundenen Zwischenrests auch Null sind, und ermöglicht es so, zu wissen, ob der Zwischenrest vom Schritt i insgesamt Null war.

Eine weitere Aufgabe der Erfindung ist die einmalige Berücksichtigung der unterschiedlichen Voraussetzungen für die Korrektur des Quotienten. Die Vorrichtung weist erfindungsgemäß Kombinatorikschaltungen für die Berechnung einer Korrektur auf, die am Ende der Division auf den Quotienten angewendet werden soll.

Vorzugsweise wird diese Korrektur erfindungsgemäß am Eingang der Speicherung der arithmetischen und logischen Einheit angewendet.

Eine weitere Aufgabe der Erfindung ist es bei jedem Divisionsschritt das Bit des komplementären Quotienten für den folgenden Schritt der Division zu berechnen, um es mit Hilfe eines Multiplexers am Eingang der arithmetischen Logikeinheit in das Bit des Quotienten mit dem niedrigsten Gewicht einzugeben.

Weitere Eigenschaften und Vorteile der Erfindung werden in der folgenden Beschreibung unter Bezugnahme auf die beiliegenden Zeichnungen, die Beispiele darstellen und die Erfindung nicht einschränken, näher erläutert, wobei:

- Fig. 1 ein Blockschema einer erfindungsgemäßen Vorrichtung zur Durchführung einer binären Division zeigt und

- Fig. 2 ein erfindungsgemäßes Funktions-Blockschema für das Erkennen von Zwischenresten mit dem Wert Null zeigt.

Ein Blockschema einer erfindungsgemäßen Vorrichtung für die Berechnung ist in Fig. 1 dargestellt.

Es weist hauptsächlich eine logische und arithmetische Verarbeitungseinheit ALU auf, die durch eine Steuereinheit 1 gesteuert wird. In der dargestellten Ausführungsform liegt eine 40 Bit-Rechenvorrichtung mit 32 Bits mit signifikanter Größe und 8 Erweiterungsbits vor.

Die Operanden am Eingang der arithmetischen Logikeinheit weisen so 8 Erweiterungsbits auf, die in der nicht mit Vorzeichen behafteten Darstellung auf Null gesetzt wurden oder in mit Vorzeichen behafteter Darstellung das Vorzeichenbit wieder aufnehmen. Das Vorzeichenbit der Operanden in der arithmetischen Logikeinheit ist also durch die Bits in der Position 31 bis 39 gegeben (bei Positionen von 0 bis 39).

Die arithmetische Logikeinheit weist zwei Eingänge Op_1 und Op_2 auf um jeweils einen rechten und einen linken Operanden zu empfangen, und einen Ausgang Op_s, um das Ergebnis einer arithmetischen oder logischen Operation auszugeben.

Die Operanden können vom mit oder ohne Vorzeichen behafteten Typ sein und ein langes (32 Bits) oder kurzes (16 Bits) Format haben.

Die Art und das Format der Operanden, die die Rechenoperationen in der arithmetischen Logikeinheit beeinflussen, sind in einem Zustandsregister RC definiert:

- SG für die mit Vorzeichen behaftete Art (1) oder nicht mit Vorzeichen behaftete Art (0) des linken Operanden und SD für den rechten Operanden.

- LG für das lange (1) oder kurze (0) Format des rechten Operanden. Für ein kurzes Format sind die Operanden linksbündig, d.h. das höchstwertigste Bit der Zahl befindet sich auf der Bitposition 31 (mit der schon gesehenen Konvention), die 16 Bits mit niedrigem Gewicht sind durch Nullen ergänzt und die 8 Erweiterungsbits mit dem Vorzeichenbit oder mit Nullen.

Die arithmetische Logikeinheit liefert auch arithmetische Informationen insbesondere die zwei höchstwertigen Übertragungsbits C_1, C_2 und das Vorzeichenbit Sr des Ergebnisses. Diese Informationsbits C_1 und Sr werden am Ende der Berechnung an das Zustandsregister RC ausgegeben. Im Beispiel wird das Bit C_1 auf das Speicherbit C des Registers und das Bit Sr auf das Vorzeichenbit S angewendet. Die Bits C und S stellen zu einem gegebenen Zeitpunkt Informationen des zuletzt berechneten Ergebnisses dar. C_1, C_2 und Sr werden auch an eine Bedingungsschaltung COND angewendet, die andere Informationsbits an das Zustandsregister RC liefert, wobei ein Bit Z_c bedeutet, daß die 32 signifikanten Bits des Ergebnisses am Ausgang der arithmetischen Logikeinheit Null sind, und ein Überlauf-Bit OVF_c, das anzeigt, daß das Ergebnis das verwendete Format übersteigt. Diese Schaltung

liefert an einem Ausgang Op_{s1} das durch die arithmetische Logikeinheit berechnete Ergebnis oder bei Übersteigen der Kapazität einen Sättigungswert, der dem Maximum oder dem Minimum des Formats entspricht. Die Bedingungsschaltung erhält auch als Informationsbit das Vorzeichenbit des linken Operanden, im Beispiel g_{39}. Diese erfindungsgemäße Bedingungsschaltung wird in bezug auf Fig. 2 später erläutert.

Erfindungsgemäß hat das Bit Z bei der Division eine andere Bedeutung: es ist das Informationsbit des Rests, der teilweise Null ist, der durch die gleiche Bedingungsschaltung COND berechnet wurde, wie es weiter unten zu sehen sein wird.

Die Operanden werden der arithmetischen Logikeinheit durch Quellenregister geliefert. Im Ausführungsbeispiel wird der linke Operand durch einen Akkumulator Acc über die Bedingungsschaltung COND (Ausgang Op_{s1}) geliefert, wobei der Akkumulator auch ein Zielregister des Ausgangs der arithmetischen Logikeinheit ist. Der rechte Operand wird durch ein Quellenregister, Reg genannt, geliefert.

Erfindungsgemäß liegt zwischen den Quellenregistern und dem Eingang Op_2 der arithmetischen Logikeinheit ein Schieberegister D und ein Multiplexer 2. Dieses Schieberegister 2 ist vorzugsweise ein Trommelregister und ermöglicht es, den Inhalt eines Quellenregisters um eine beliebige Anzahl von Positionen nach rechts oder nach links arithmetisch oder logisch zu verschieben, bevor er an den Eingang eines Operanden der arithmetischen Logikeinheit angelegt wird, im Beispiel der linke Operand. Erfindungsgemäß könnte es sich auch um ein einfacheres Schieberegister handeln, das es ermöglicht, wenigstens eine Verschiebung einer Position nach links auszuführen.

Die Ausgangsbits dieses Schieberegisters, bd_0-bd_{39} genannt, werden auf die Bits g_0-g_{39} des Eingangs des linken Operanden Op_2 angewandt. Genauer gesagt werden die Bits bd_1-bd_{39} direkt auf die entsprechenden Bits des linken Operanden angewandt. Erfindungsgemäß wird das Bit bd_0 an den Eingang des logischen Multiplexers 2 angelegt, der an den anderen Eingängen ein Vergleichsbit für das Vorzeichen CS und ein Bit des komplementären Quotienten NQ empfängt, die vom Zustandsregister RC kommen. Der Multiplexer liefert ein Bit am Ausgang, das auf das Bit g_0 des linken Operanden angewendet wird.

Um erfindungsgemäß eine Division auszuführen, wird der Zähler im Akkumulator am Eingang der arithmetischen Logikeinheit mit einem Schieberegister, der linke Operand im Beispiel, und der Nenner im Quellenregister auf den anderen Operanden, rechts im Beispiel, gesetzt.

Das Vergleichsbit für das Vorzeichen CS des Zustandsregisters RC kommt erfindungsgemäß von unterschiedlichen Bedingungsschaltungen, die für die Division verwendet werden, und kann nach dem jeweiligen Divisionsschritt das Vergleichsbit für das Vorzeichen des Dividenden und des Divisors oder ein Korrekturbit des Quotienten darstellen.

Eine erste Bedingungsschaltung 3 liefert ein erstes Bit CS_1, das das unterschiedliche ($CS_1 = 1$) oder gleiche ($Cs_1 = 0$) Vorzeichen des Nenners (rechter Operand) und des Dividenden (linker Operand) anzeigt und das verwendet werden wird, um das erste Bit des Quotienten (Vorzeichenbit) zu bestimmen und um die mögliche Korrektur des Quotienten zu berechnen. In der in Fig. 1 gezeigten Ausführungsform besteht sie aus einem Exklusiv-ODER-Gatter, das am Eingang das Vorzeichenbit des linken Operanden und das Vorzeichenbit des rechten Operanden empfängt. Es liefert am Ausgang

ein erstes Vorzeichenvergleichsbit, genannt CS_1. In dem gewählten Fall einer Rechenvorrichtung mit Erweiterungsbits ist das Vorzeichenbit durch eine signierte Zahl durch das Bit mit höchstem Gewicht (hier Position 31) und alle Erweiterungsbits (die eine Kopie dieses Bits 31 sind) gegeben. In dem Fall einer Zahl ohne Vorzeichen ist das Vorzeichen positiv, und das Bit mit dem höchsten Gewicht (Position 31) stellt nicht mehr das Vorzeichen dar, sondern eine Größe der dargestellten Zahl, wobei die Zahl sich nun mit Nullen bis zum Bit 39 erstreckt. Wenn das Bit CS_1 in allen Fällen korrekt gesetzt werden soll, wählt man ein Erweiterungsbit, beispielsweise die Bits der Positionen 39 des linken und rechten Operanden. Man kann ebenso eine UND-Logik zwischen dem Operanden mit oder ohne Vorzeichen und dem Positionsbit 31 verwenden, um das wirkliche Vorzeichen des Operanden in allen Fällen zu erhalten. Dies ist mit einem UND-Gatter 4 dargestellt, das das Vorzeichen des Nenners für den rechten Operanden, Sdiv genannt, aufgrund des Bits der Art SD des Zustandsregisters und des Bits d_{31} liefert.

Erfindungsgemäß liefert eine zweite Bedingungsschaltung 5 ein zweites Bit CS_2, das die Korrektur darstellt, die auf den Quotienten am Ende der Division angewendet werden muß.

Sie weist im Beispiel drei logische Gatter 6, 7 und 8 auf. Sie empfängt am Eingang die Bits Z, CS und das Bit Sdiv, das, wie gesehen, das Vorzeichen des Nenners anzeigt, ob es ein Vorzeichen hat oder nicht. Das Bit Z stellt erfindungsgemäß in diesem Fall das Informationsbit des Restes, der teilweise Null ist, dar.

Das Gatter 6 ist ein logisches UND-Gatter, das an den Eingängen das Bit Z und das Bit Sdiv für das Vorzeichen des Nenners empfängt.

Das Gatter 7 ist ein logisches UND-Gatter, das das gleiche Bit Z an einem invertierenden Eingang und das Vergleichsbit für das Vorzeichen CS des Registers RC an einem normalen Eingang empfängt. Das Gatter 8 ist ein ODER-Gatter, das an den Eingängen die Ausgänge der vorherigen Gatter 6 und 7 empfängt. Es liefert ein logisches Ausgangsbit, das das zweite Korrekturbit CS_2 des Quotienten darstellt.

Ein drittes Bit CS_3 ist vorgesehen, welches es der Steuereinheit 1 anstelle der Schaltung 3 (dies in den Fällen wo das Vorzeichen des Nenners und des Zählers vorher bekannt sind) ermöglicht, direkt zu schreiben, ob der Zähler und der Nenner von gleichem Vorzeichen sind oder nicht.

Die drei Bits CS_1, CS_2 und CS_3 werden an einen Multiplexer 9 angelegt, dessen Ausgang es ermöglicht, das eine oder andere Bit auf das Bit CS_c umzuschalten, das in dem Bit CS des Zustandsregisters RC gespeichert ist.

Die arithmetische Logikeinheit empfängt ebenso ein Bit Cin, welches eine Speicherung auf dem Bit mit dem niedrigsten Gewicht ermöglicht oder nicht. Im allgemeinen und wie dargestellt, empfängt ein Multiplexer Mux am Eingang das Übertragungsbit C des Zustandsregisters und ein auf Null gesetztes Bit, um den einen oder anderen Eingang anzulegen, je nachdem, ob die Operationen mit oder ohne Übertrag in der arithmetischen Logikeinheit ausgeführt werden.

Erfindungsgemäß ist ein weiterer Eingang auf dem Multiplexer Mux vorgesehen, um das Korrekturbit CS_2 zu empfangen, das durch die vorher beschriebene Bedingungsschaltung 5 berechnet wurde.

Das Bit des komplementären Quotienten NQ des Zustandsregisters wird durch eine Bedingungsschaltung 10 geliefert, die

das Vorzeichen des Nenners mit dem Vorzeichen des Rests vergleicht (nach der Subtraktion oder Addition des Nenners). Das Vorzeichen des Nenners ist, wie gezeigt, durch das Bit Sdiv gegeben, und das Vorzeichen des hier betroffenen Rests wird durch das Vorzeichenbit des Ergebnisses geliefert, das durch die arithmetische Logikeinheit berechnet wurde, d.h. das Bit Sr, das im allgemeinen dem Bit von Position 39 des Ergebnisses (b_{39}) entspricht.

In der in Fig. 1 dargestellten Ausführungsform weist sie weiterhin ein Exklusiv-ODER-Gatter (XOR) auf, das am Eingang das Bit Sdiv und das Bit Sr empfängt. Der Ausgang dieses Gatters 11 liefert das Bit des komplementären Quotienten, das in dem Bit NQ des Zustandsregisters RC gespeichert ist.

Die erfindungsgemäße Bedingungsschaltung COND wird nun in bezug auf Fig. 2 genauer beschrieben. Sie weist einen ersten Teil auf, der Schaltungen 12 bis 18 aufweist, um das Bit Z zu bilden, das erfindungsgemäß entweder ein Ergebnis Null oder Information des Restes, der teilweise Null ist, darstellt, und einen zweiten Teil, der die Schaltungen 19 bis 22 aufweist, um das Bit OVF zu bilden, das erfindungsgemäß entweder einen Überlauf oder eine unmögliche Division darstellt.

Der erste Teil empfängt die 16 Bits mit niedrigstem Gewicht b_0 - b_{15} in einer ersten Schaltung 12 für den Vergleich mit Null, die ein Ausgangsbit Zl gleich 1 liefert, wenn alle Eingangsbits gleich Null sind, und Null, wenn nicht. Die 16 folgenden Bits b_{16} bis b_{31} werden an den Eingang einer zweiten Schaltung 13 für den Vergleich mit Null angelegt. Diese liefert ein Ausgangsbit Zh gleich 1, wenn alle Eingangsbits b_{16} bis b_{31} gleich Null sind, und gleich Null, wenn nicht. Die Schaltungen 12 und 13 sind auf einfache Weise mit Hilfe

von vier NICHT-ODER-Gattern ausgeführt, eines für vier folgende Bits eines UND-Gatters mit vier Eingängen (nicht dargestellt).

Die zwei Bits Zh und Zl werden an den Eingang eines logischen UND-Gatters 14 angelegt, das ein erstes Bit Z_1 liefert, das anzeigt, ob die signifikanten Bits des Ergebnisses b_0 bis b_{31}, die durch die arithmetische Logikeinheit berechnet wurden, alle Null sind ($Z_1 = 1$) oder nicht ($Z_1 = 0$).

Erfindungsgemäß arbeitet die Bedingungsschaltung ein anderes Bit Z_4 mit Hilfe von logischen Schaltungen 15 bis 17 aus.

Ein logisches UND-Gatter empfängt das Bit Zh an einem Eingang und das Bit b_{15} an einem invertierenden Eingang. Es liefert einen logischen Ausgang Z_2 gleich 1, wenn die Bits b_{15} und b_{16} bis b_{31} alle Null sind, gleich Null, wenn nicht.

Ein logisches UND-Gatter 16 empfängt am Eingang das Bit Z des Zustandsregisters RC und das Bit b_{15} des Ergebnisses am Ausgang der arithmetischen Logikeinheit an einem invertierenden Eingang. Es liefert einen logischen Ausgang Z_3, der an den Eingang eines ODER-Gatters 17 angelegt wird, das an einem zweiten Eingang den Ausgang Z_2 des Gatters 15 empfängt. Dieses Gatter 17 liefert am Ausgang ein Ergebnisbit, das auf 1 gesetzt ist, wenn die Bits b_{15} und b_{16} bis b_{31} gleich Null sind oder wenn das Bit Z des Zustandsregisters gleich 1 ist und damit das Bit b_{15} Null ist. Das Bit Z_4 ist im anderen Fall gleich Null.

Der Ausgang Zl der ersten Vergleichsschaltung 12 wird ebenso verwendet, um anzuzeigen, ob der Quotient als Ergebnis der Division, der in den 16 Bits mit dem geringsten Gewicht

ausgedrückt ist, Null ist oder nicht, was es ermöglicht, den Rest der Division zu korrigieren. Diese drei Bits Z_1, Z_4 und Z1 werden an den Eingang eines Multiplexers 18 angelegt, der durch den Zustand des Bits Z_c gesteuert wird, das in dem Bit Z des Zustandsregisters RC gespeichert ist.

Die Bedingungsschaltung weist weiterhin eine Sättigungsschaltung 19 auf, die hauptsächlich von der arithmetischen Logikeinheit die Übertragungsbits C_1 und C_2 mit höchstem Gewicht und das Vorzeichenbit Sr erhält, um festzustellen, ob ein Überlauf vorliegt, und liefert ein entsprechendes Informationsbit OVF_1 und eventuell am Ausgang einen entsprechenden Sättigungswert (Op_{s1}).

Erfindungsgemäß weist sie weiterhin eine logische Schaltung auf, um festzustellen, ob die Division mit einem Zähler und einem Nenner am Eingang der arithmetischen Logikeinheit möglich ist oder nicht, d.h., ob der absolute Wert des Zählers auf jeden Fall kleiner ist als der absolute Wert des Nenners (indem die Zahlen betrachtet werden wie sie im Bruchformat ausgedrückt sind). Das erfolgt dadurch, daß festgestellt wird, ob das Ergebnis einer Subtraktion, wenn die zwei Operanden das gleiche Vorzeichen haben, oder einer Addition, wenn sie ein entgegengesetztes Vorzeichen haben, bestimmt negativ ist oder nicht.

In dem Beispiel weist sie ein exklusives NICHT-ODER-Gatter 20 auf, das an den Eingängen die Bits Sr des Ergebnisses am Ausgang der arithmetischen Logikeinheit erhält, die die Addition oder die Subtraktion ausgeführt hat, und das Vorzeichenbit des in dem Beispiel linken Operanden g_{39} und ein zweites logisches ODER-Gatter, das am Eingang den Ausgang des Gatters 20 und das erste Bit Z_1, das ein Ergebnis Null anzeigt, empfängt. Der Ausgang des Gatters 21 liefert eine Information OVF_2, welche anzeigt, daß eine Division des

linken Operanden durch den rechten Operanden nicht möglich ist, wenn sie 1 ist, wobei der Zähler nicht absolut kleiner als der Nenner (in Absolutwerten) ist.

Ein Multiplexer 22 ist vorgesehen, der das eine (OVF_1)- oder das andere (OVF_2)-Bit auf das Bit OVF_c umschaltet, das in dem Bit OVF des Zustandsregisters RC gespeichert ist.

Alle unterschiedlichen beschriebenen Schaltungen werden durch die Steuereinheit 1 gesteuert:

- die arithmetische Logikeinheit durch einen Befehl mALU,

- die Bedingungsschaltung durch einen Befehl $mCOND_1$, der den Multiplexer 18 des Schaltkreises für die Erzeugung des Bits Z_c steuert und durch einen Befehl $mCOND_2$, der den Multiplexer 22 des Schaltkreises für das Erzeugen des Bits OVF_c steuert,

- das Zustandsregister RC durch einen Befehl mRC, der es der Steuereinheit ermöglicht, alle oder einen Teil der vorher beschriebenen Zustandsbits in dieses Register zu schreiben,

- das Schieberegister D durch einen Befehl mD,

- der Multiplexer 2 am linken Eingang der arithmetischen Logikeinheit durch einen Befehl m2,

- der Multiplexer Mux durch den Befehl mMux, um eine Speicherung in der arithmetischen Logikeinheit zu ermöglichen,

- der Multiplexer 9 durch einen Befehl m9, um das Bit CS_c an das Zustandsregisterbit CS zu schicken.

Der Steuervorgang der erfindungsgemäßen Vorrichtung kann also mit dem Nenner im Quellenregister, auf dem rechten Operanden, und den Zähler im Akkumulator, auf dem linken Operanden, beschrieben werden.

Die Steuereinheit positioniert die Bits SD und SG, die die signierte Art der rechten und linken Operanden ausdrücken. Es genügt, eines davon, beispielsweise das des Nenners zu positionieren, da man nur mit Operanden von gleicher Art arbeiten will.

In der Darstellung der Fig. 1 und 2 mit einer arithmetischen Logikeinheit von 40 Bits mit 8 Erweiterungsbits ist das Format der Operanden vorzugsweise wie folgt: Der Zähler wird in 32 Bits in einem langen Format dargestellt (Bit von langem Format LG, das in dem Zustandsregister gesetzt wird).

Der Nenner ist eine Größe, die in 16 Bit ausgedrückt wird. Das Bit von kurzem Format LD wird im Zustandsregister gesetzt. Es wird in dem Register Reg gespeichert.

Es wird angenommen, daß eine Division mit Vorzeichen ausgeführt wird (SD = SG - 1). Auf Befehl der Steuereinheit, die zuvor die Bits SD, LD, SG und LD gelesen hat, wird der Zähler am linken Eingang der arithmetischen Logikeinheit mit dem Vorzeichenbit auf dem Bit mit Position 31 und die Vorzeichenerweiterung auf den Bits 32 bis 39 geladen. Der Nenner wird am rechten Eingang der arithmetischen Logikeinheit in den 16 Bits 16 bis 31 mit hohem Gewicht mit den 16 Bit mit niedrigem Gewicht (Position 0 bis 15), die auf Null gesetzt sind (kurzes Format). Das Vorzeichen belegt die Bits der Position 32 bis 39 (mit Vorzeichen).

Erfindungsgemäß wird eine erste Vergleichsinstruktion des Vorzeichens ausgeführt, wenn das jeweilige Vorzeichen der Operanden (Zähler und Nenner) nicht bekannt ist, deren Ziel es ist, zu bestimmen, ob die Operanden das gleiche Vorzeichen haben oder nicht. Die Steuereinheit empfängt also einen Vergleichsbefehl CMPS für das Vorzeichen, wodurch der Multiplexer 9 (Befehl m9) angewiesen wird, das Bit CS_1 auf den Ausgang CS_c des Multiplexers umzuschalten. Das Bit CS_1 wird durch das Exklusiv-ODER der Vorzeichen der Operanden geliefert. Wenn das Vorzeichen der Operanden bekannt ist, kann die Steuereinheit das Bit CS_3 direkt auf das Bit CS_c umschalten. Das Bit CS_c wird also im Bit CS des Zustandsregisters RC gespeichert.

Nachfolgend wird festgestellt, ob die Division möglich ist. Für die Durchführung des Divisionsalgorithmus ohne Berücksichtigung der Zwischenreste ist es nötig, daß der Absolutwert des Zählers auf jeden Fall kleiner als der Absolutwert des Nenners ist (unter der Bedingung, daß die Zahlen im Bruchformat angegeben sind). Erfindungsgemäß wird das Bit für den Überlauf OVF im Zustandsregister verwendet, um die Ungültigkeit der Division anzuzeigen, wird aber wie gesehen auf andere Art für die Division berechnet (OVF_2, Fig. 2, Schaltungen 20 und 21).

Die Steuereinheit, die den entsprechenden Befehl CHKDIV empfängt, liest das Vergleichsbit für das Vorzeichen CS des vorher gesetzten Zustandsregisters RC, und wenn der Wert 1 ist, was entgegengesetzte Vorzeichen der Operanden anzeigt, befiehlt sie der arithmetischen Logikeinheit, die beiden Operanden zu addieren, und wenn ihr Wert Null ist, den Nenner (rechter Operand) vom Zähler (linker Operand) abzuziehen. Diese Operationen erfolgen ohne Ausgabe des Resultats an den Akkumulator, der unverändert bleibt. Anschließend schickt die Steuereinheit an den Multiplexer 22 der Bedin-

gungsschaltung COND einen Befehl $mCOND_2$ zum Umschalten des Bits OVF_2 in das Bit OVF_C, was in dem Bit OVF des Zustandsregisters gespeichert wird. Es ergibt sich also:

$$OVF_C = OVF_2 = !(Sr \oplus g_{39}) + Z_1$$

(das ! deutet an, daß das Komplement verwendet werden muß).

Das im Register RC gespeicherte Bit OVF ist also gleich 1, wenn das Ergebnis in den Bits 0 bis 31 Null ist, oder wenn das Vorzeichen des Ergebnisses, das durch das Bit SR am Ausgang der arithmetischen Logikeinheit gegeben ist (und gleich in der Praxis zum Bit b_{39} des Ergebnisses ist) und das Vorzeichen des Zählers gleich sind.

Wenn die Division möglich ist (OVF = 0), kann schließlich der erste Schritt der Division ausgeführt werden. Erfindungsgemäß besteht der erste Schritt der Division darin, das Vorzeichenbit des Quotienten, den neuen Zwischenrest (den ersten Zwischenrest) und das Bit des komplementären Quotienten NQ für den folgenden Schritt zu berechnen. Hierfür wird ein Befehl DIVS verwendet, der ein einziges Mal verwendet wird, um die Division mit Vorzeichen auszuführen. Es bleibt festzuhalten, daß das Vorzeichenbit des Quotienten, das durch das Bit CS gegeben ist und das durch CS_1 oder direkt durch die Steuereinheit gesetzt wurde, bekannt ist.

Beim Empfang des Befehls DIVS führt die Steuereinheit folgende Befehle aus:

- Abschicken eines Befehls mD an das Schieberegister, damit der Inhalt des Akkumulators (der noch der Dividend ist) um eine Position nach links verschoben wird.

- Abschicken eines Befehls m2 an den Multiplexer 2 am Eingang des linken Operanden, damit er das Zustandsregisterbit CS auf das Bit aus der Position mit dem geringsten Gewicht des linken Operanden umschaltet. Es liegt also am linken Eingang der arithmetischen Logikeinheit der mit dem Bit CS auf sein Bit g_0 um ein Bit nach links verschobene Dividend: Das Bit g_0 ist der Zwischenquotient des ersten Divisionsschrittes, der das Vorzeichenbit darstellt.

- Abschicken eines Befehls mALU an die arithmetische Logikeinheit, damit der Nenner vom Rest abgezogen wird, wenn das Bit CS gleich Null ist, oder addiert wird, wenn das Bit CS gleich 1 ist, wobei das Ergebnis im Akkumulator festgehalten wird. 16 Bits mit der Wertigkeit 0 bis 15 des Nenners sind Null (kurzes Format): einzig die Bits mit der Wertigkeit 16 bis 39 sind von dieser Operation betroffen, die Bits mit der Wertigkeit 0 bis 15 des linken Operanden bleiben unverändert. Man hat so den Zwischenrest berechnet.

- Speichern des komplementären Quotientenbits NQ in dem Zustandsregister RC, gegeben durch:
  NQ = (Sdiv) XOR Sr,
  d.h., daß das Vorzeichenbit bei Division mit Vorzeichen (SD = 1) durch das Exklusiv-ODER zwischen dem Vorzeichenbit des Nenners und dem Vorzeichenbit des neuen berechneten Zwischenrests SR (Bit b_{39}) gegeben ist.

- Abschicken eines Befehls $mCOND_1$ an den Multiplexer 18 der Bedingungsschaltung, um das Bit Z_4 des Rests, der teilweise Null ist, auf das Bit Z_C umzuschalten, das in dem Bit Z des Zustandsregisters RC gespeichert ist und erfindungsgemäß den Rest, der teilweise Null ist, bei diesem ersten Divisionsschritt aufzeigt.

Man kann also zu den folgenden Divisionsschritten kommen. Wenn einmal der Divisionsschritt der Berechnung des Bits des Vorzeichens des Quotienten ausgeführt wurde, müssen die folgenden Bits des Quotienten berechnet werden.

Erfindungsgemäß wird hierfür ein DIVQ genannter Befehl verwendet, der den neuen Zwischenquotienten berechnet, anschließend den neuen Zwischenrest, anschließend das Bit des komplementären Quotienten NQ für den folgenden Schritt, und schließlich speichert man das Auftreten eines Restes, der teilweise Null ist, oder die Bestätigung eines Restes, der teilweise Null ist.

Bei dem erfindungsgemäßen Befehl DIVQ führt die Steuereinheit nun folgende Operationen aus:

- Abschicken eines Befehls mD an das Schieberegister, um den Inhalt des Akkumulators um eine Position nach links mit Transfer zum linken Operanden zu schieben.

- Abschicken eines Befehls m2 an den Multiplexer 2, um die Invertierung des Bits NQ des Zustandsregisters auf das Bit g_0 mit niedrigstem Gewicht des linken Operanden umzuschalten.

- Abschicken eines Befehls mALU an die arithmetische Logikeinheit, um die Subtraktion oder Addition des Nenners und des so berechneten linken Operanden, je nachdem ob der Wert des Bits NQ des Bedingungsregisters 0 oder 1 ist, auszuführen unter Beibehaltung des Ergebnisses im Akkumulator.

- Berechnung des Bits des komplementären Quotienten NQ in dem Zustandsregister für den folgenden Divisionsschritt, wie vorher gesehen.

- Aktualisierung des Bits Z des Zustandsregisters mit dem Informationsbit Z_4 für den Rest, der teilweise Null ist, indem ein Umschaltbefehl $mCOND_1$ an den Multiplexer 18 der Bedingungsschaltung gesendet wird.

Erfindungsgemäß wird einerseits getestet, ob die Bits 15 und 16 bis 31, die die höherwertigen Bits des Rests sind, Null sind, um die Information des Restes, der teilweise Null ist, auf 1 zu setzen, und es wird andererseits das Bit 15 mit dem Rest, der teilweise Null ist, getestet, das schon auf 1 gesetzt wurde, um nach und nach festzustellen, ob der Inhalt das Akkumulationsregister nach links verschoben ist, wenn die anderen Bits des jeweiligen Rests mit geringerem Gewicht, die nach und nach zu denen mit höherem Gewicht verschoben sind, ebenfalls Null sind. Wenn dies nicht der Fall ist, wird das Informationsbit des Restes, der teilweise Null ist, auf Null zurückgesetzt. Zusammenfassend wird, wenn im Schritt i ein Rest der teilweise Null ist gefunden wird, anschließend bei jedem weiteren Schritt getestet, ob das auf das Bit 15 verschobene Bit mit niedrigerem Gewicht Null ist oder nicht. Wenn bei einem Schritt i + k ein Bit 15 ungleich Null gefunden wird, bedeutet dies, daß der Rest im Schritt i nicht gänzlich Null war. Parallel zu diesem Test wird weiterhin bei jedem Schritt getestet, ob der neu berechnete Zwischenrest teilweise Null ist oder nicht. Am Ende der Division zeigt das Bit Z an, ob es einen Zwischenrest gab, der teilweise Null war oder nicht. Das Erkennen geschieht also während der ganzen Division mit einer Speicherung dessen, was geschieht, und das Erkennen wird am Ende der Division interpretiert. Es wird somit auf vernünftige Weise ein Operand von variabler Größe

auf Null überprüft. Es ist festzuhalten, daß nur ein Rest Null im Laufe einer Division auftreten kann.

Wenn ein Divisionsschritt DIVS und vierzehn Divisionsschritte DIVQ ausgeführt wurden, gibt der letzte Befehl DIVQ das letzte Bit Z aus, welches anzeigt, ob ein Zwischenrest Null vorlag oder nicht.

Es muß also noch der Zwischenquotient korrigiert werden, um den endgültigen Quotienten zu erhalten. Erfindungsgemäß werden zugleich alle Bedingungen berücksichtigt, bei denen eine Einheit auf der Position mit niedrigerem Gewicht addiert werden muß. Erfindungsgemäß wird die Bedingungsschaltung 5 verwendet, um ein Kumulationsbit zu berechnen, das CS_2 darstellt, und man verwendet den Eingang der Speicherung Cin der arithmetischen Logikeinheit, um dieses Bit CS_2 auf der Position mit niedrigstem Gewicht unter Beibehaltung des Ergebnisses im Akkumulator zu addieren. Ein Ausführungsbeispiel der Bedingungsschaltung wurde im Zusammenhang mit Fig. 1 beschrieben. Die Kumulationsbedingung lautet also: $CS_2 = (Sdiv.Z) + \overline{Z}/CS)$.

Dieses Bit vereint die beiden Fälle, in welchen der Quotient korrigiert werden muß: erstens wenn der Zähler und der Nenner ein entgegengesetztes Vorzeichen haben, was durch das Bit CS des Zustandsregisters angezeigt wird, und zweitens, wenn der Nenner negativ ist und wenn wenigstens ein Zwischenrest aufgetreten ist, der während der Division insgesamt Null war: in diesem Fall wurde der Zwischenrest als positiv betrachtet, was zu einem Bit NQ gleich 1 führt, d.h. ein Bit im Zwischenquotienten gleich Null im folgenden Schritt. Ein Rest Null muß betrachtet werden, als wenn er das gleiche Vorzeichen wie der Nenner hätte, und so wird für den folgenden Schritt ein Bit in dem Quotienten gleich 1 eingeführt (NQ gleich 0).

In diesen beiden Fällen verbessert die Nachinkrementierung des Resultats den Quotienten.

Die Steuereinheit, die den Befehl empfängt, den Quotienten RESQ zu verbessern, führt also folgende Befehle aus:

- Abschicken eines Befehls mD an das Schieberegister D, damit es als linken Operanden den Inhalt des um eine Position nach links verschobenen Akkumulators auflädt.

- Abschicken eines Befehles m2 an den Multiplexer 2, damit er das Inverse des Bits NQ auf das Bit des linken Operanden mit dem niedrigsten Gewicht umschaltet.

- Abschicken eines Befehls mMux an den Multiplexer Mux, damit das Bit CS_2 der Bedingungsschaltung 5 an den Eingang der Speicherung Cin der arithmetischen Logikeinheit umgeschaltet wird.

- Abschicken eines Befehls mALU an die arithmetische Logikeinheit, damit sie das Gespeicherte dem linken Operanden hinzufügt und das Ergebnis im Akkumulator behält. Im Akkumulator liegt also der richtige Quotient im Format mit Vorzeichen auf den 16 Bits mit niedrigerem Gewicht vor. Der Rest wird in den 24 anderen Bits ausgedrückt.

- Abschicken eines Befehls m9 an den Multiplexer 9, um das Bit CS_2 auf das Bit CS des Bedingungsregisters umzuschalten.

- Abschicken eines Befehls mCOND1 an den Multiplexer 18 der Bedingungsschaltung 5, um das Bit Z1 auf das Bit Z des Zustandsregisters umzuschalten, welches anzeigt, ob der korrigierte Quotient Null ist oder nicht.

Der erfindungsgemäße Befehl RESQ berechnet also die neuen Bits Z und CS. Das Bit CS bleibt unverändert, wenn kein Zwischenrest entdeckt wurde. Das Bit Zl ist 1, wenn die 16 niedrigeren Bits (Quotient) des Ergebnisses der Operation in der arithmetischen Logikeinheit Null sind.

Diese Voraussetzung dient dazu, den Rest in dem Fall zu verbessern, in dem der Quotient vor dem Befehl nur Einsen trägt und CS_2 1 ist: Der Quotient ist also Null, aber die Fortpflanzung des Gespeicherten hat zur Folge, daß der Zwischenrest verändert wird.

Die Aktualisierung des Bits CS mit dem Bit CS_2 dient auch dazu, den Zwischenrest zu verbessern.

Erfindungsgemäß wird ein Korrekturbefehl des Zwischenrestes RESR ausgeführt. Die Steuereinheit führt Tests an den Bits CS, NQ und Z des Zustandsregisters aus. Wie gesehen ist das Bit Z durch Zl gegeben und dient dazu, einen Fehler aufgrund der Korrektur des Quotienten zu korrigieren.

Das Bit CS selbst enthält das Korrekturbit CS_2 des Quotienten. Das Bit NQ enthält das letzte Bit des durch den letzten Befehl DIVQ berechneten komplementären Quotienten (der Befehl RESQ verändert nicht das Bit NQ, die Steuereinheit schickt keinen Befehl mRC des Bits NQ für diesen Befehl an das Zustandsregister).

Wenn CS = NQ gilt, führt die Steuereinheit nichts aus.

Wenn CS = 0 und NQ = 1 gilt, addiert die Steuereinheit zu dem Nenner den Rest. Für diesen Vorgang verschiebt sie vorzugsweise das Register Reg, das den Nenner enthält, über das Schieberegister, an das sie einen Befehl mD schickt, um

eine Position nach links, um zweimal den Nenner auf dem linken Operanden zu erhalten, und schickt den Inhalt des Akkumulators auf den rechten Operanden (gestrichelte Pfeile).

Wenn CS = 1 und NQ = 0 ist, zieht die Steuereinheit zweimal den Nenner vom Akkumulator ab. In diesem letzten Fall läßt die Steuereinheit die arithmetische Logikeinheit die Summe des Akkumulators (rechter Operand) und des 2-Komplements des Zweifachen des Nenners (linker Operand) berechnen, wenn Z = 0 ist, oder des 1-Komplements des Zweifachen des Nenners, wenn Z = 1 ist (in diesem Fall wurde die Inkrementierung "+1", die in der 2-Komplementberechnung vorlag, schon im Befehl RESQ ausgeführt).

Das Ergebnis der arithmetischen Logikeinheit wird also im Akkumulator geladen, aber nur auf den 24 höchsten Bits. Der Quotient bleibt auf den 16 niedrigsten Bits unverändert, und der wahre Rest befindet sich also in den 24 höchsten Bits.

Die Division mit Vorzeichen, die im Zusammenhang mit der erfindungsgemäßen Anordnung zur Durchführung gerade erklärt wurde, betrifft Operanden im Bruchformat. Wenn die Operanden ganzzahlig sind, wird zuerst der Dividend um eine Position nach links verschoben, nachdem der Vergleich des Vorzeichens (CS) durchgeführt wurde. Der Rest am Ende der Division muß als das Doppelte des wahren Restes interpretiert werden.

Die Überprüfung, ob die Division möglich ist, wird nach dem Verschieben des Dividenden ausgeführt.

Was die Division ohne Vorzeichen betrifft, entfällt das Vorzeichenbit. Es muß dennoch erfindungsgemäß ein Befehl

DIVS für die Berechnung des Vorzeichens des Quotienten ausgeführt werden, um das erste Bit des Quotienten zu berechnen. Dieser Befehl erzeugt ein erstes, nicht bedeutsames Bit im Quotienten. Anschließend müssen 15 Befehle DIVQ ausgeführt werden, um die anderen Bits des Quotienten zu berechnen. Es liegt also ein zusätzlicher Befehl DIVQ vor. Der Quotient wird also in 17 Bits ausgedrückt, aber das Bit mit der Wichtung 16, das das Vorzeichenbit darstellt, ist nicht von Bedeutung. Der Rest wird also auf den 23 Bits mit dem höchsten Gewicht dargestellt.

Das Wechseln der Divisionsart von Division mit Vorzeichen zu Division ohne Vorzeichen wird durch Veränderung des Bits SD im Zustandsregister ausgeführt: SD = 0 zeigt an, daß der Nenner kein Vorzeichen hat und somit Sdiv = 0 ist. Für den Dividenden genügt es, daß die 8 Erweiterungsbits Nullen enthalten.

Für den Fall der Division ohne Vorzeichen gibt es keinen Unterschied, ob das Format der Operanden ganzzahlig oder ein Bruch ist, was nur eine Rolle in der Interpretierung des abschließenden Ergebnisses spielt.

Mit der erfindungsgemäßen Vorrichtung zur Durchführung und den verschiedenen Kombinatorik- und Multiplexschaltungen wird vorzugsweise jeder der verschiedenen Befehle in einer einzigen Abfolge ausgeführt. Ein Anwender muß keine Interrupt-Routinen durchführen, um das Ergebnis zu korrigieren, nur in dem Fall, in dem die Division unmöglich ist (OVF_2).

In dem Beispiel einer Division mit Vorzeichen von Bruchzahlen benötigt eine Division mit Vorzeichen von 32 Bits durch 16 Bits 18 elementare Befehle (CMPS, CHKDIV, DIVS, 14 mal DIVQ, RESQ) plus vielleicht einen weiteren, wenn der Rest korrigiert werden soll für den Fall, daß man ihn für andere

Rechnungen benötigt (es muß auch ein Befehl zur Initialisierung des Taktzählers DIVQ angehängt werden). Der Quotient wird am Ende in 16 Bits dargestellt.

Man kann sicherlich weniger Bits des Quotienten wählen (also weniger Divisionsschritte DIVQ), wenn eine so große Genauigkeit nicht nötig ist.

Ansprüche

1. Vorrichtung zur digitalen Durchführung einer Divisionsoperation nach der Methode der Zwischenrestvernachlässigung, die umfaßt:
eine arithmetische Logikeinheit zur Berechnung des Zwischenrests,
eine Schaltung zum Erkennen von Zwischenresten Null im Lauf der Division,
einen mit dem Dividenden initialisierten Akkumulator, der anschließend für jeden Divisionschritt den Rest auf den Bits mit dem höchsten Gewicht und den Quotient auf den Bits mit dem niedrigsten Gewicht enthält,
ein Zustandsregister (RC), das Informationsbits über die letzte ausgeführte arithmetische oder logische Operation enthält,
ein Quellenregister (Reg) am Eingang für einen rechten Operanden (Op_1) der arithmetischen Logikeinheit für die n Bits des Nenners,
dadurch gekennzeichnet, daß
sie ein Schieberegister (D) am Eingang für den linken Operanden (Op_2) der arithmetischen Logikeinheit aufweist, um den Inhalt des Akkumulators um eine Position nach links zu verschieben, und einen Multiplexer (2), um beim ersten Berechnungsschritt des Quotienten (DIVS) ein Vergleichsbit der Vorzeichen des Nenners und des Zählers (CS_1) zu liefern, das in einem Bit (CS) des Zustandsregisters gespeichert ist und das das Vorzeichenbit des Quotienten darstellt, und um als Bit mit niedrigstem Gewicht des linken Operanden bei den folgenden Schritten der Berechnung des Quotienten (DIVQ) ein komplementäres Quotientenbit (NQ) zu liefern, das im vorherigen Schritt

i-1 berechnet wurde und im Zustandsregister gespeichert wurde, das das Bit mit dem niedrigsten Gewicht des Quotienten beim Schritt i der Berechnung des Quotienten darstellt, und daß bei jedem Berechnungsschritt des Quotienten ein Null-Erkennungsbit, das durch die Schaltung für das Erkennen von Zwischenresten Null berechnet wurde, mit einem Bit Z des Zustandsregisters belegt wird.

2. Vorrichtung nach Anspruch 1, dadurch gekennzeichnet, daß die Schaltung für das Erkennen eines Zwischenrestes Null Kombinatorikschaltungen aufweist, um ein Null-Erkennungsbit bei jedem Schritt i der Division zu berechnen, um ein Bit (Z) des Zustandsregisters zu aktualisieren, so daß das im Schritt i berechnete Bit gleich 1 ist, wenn alle Bits b_{16} bis b_{31} und das Bit b_{15} des im Akkumulator gespeicherten Ergebnisses Null sind, oder wenn das im vorherigen Schritt i-1 berechnete und im Zustandsregister gespeicherte (Z) Erkennungsbit gleich 1 ist und das Bit b_{15} gleich Null ist.

3. Vorrichtung nach Anspruch 2, dadurch gekennzeichnet, daß sie eine Schaltung zur Korrektur des Quotienten, die eine Kombinatorikschaltung aufweist, um ein Korrekturbit (CS_2) des Quotienten zu berechnen, so daß dieses Bit (CS_2) gleich 1 ist, wenn der Nenner negativ ist (Sdiv = 1) und die Detektion eines Restes, der gänzlich Null war (Z = 1), erfolgte oder kein Erkennen eines Restes, der gänzlich Null war, erfolgte (!Z = 1), aber der Nenner und der Dividend ein unterschiedliches Vorzeichen haben ($CS = CS_1 = 1$), und einen Multiplexer (MUX) um das Fehlerkorrekturbit (CS_2) als Übertragsbit (Cin) am Eingang der arithmetischen Logikeinheit anzulegen.

4. Vorrichtung nach Anspruch 3, dadurch gekennzeichnet, daß das Bit zur Korrektur des Quotienten anschließend in ei-

nem Bit (CS) des Zustandsregisters gespeichert wird und ein Bit (Z1), das anzeigt, ob alle Bits b_0 bis b_{15} des verbesserten Quotienten Null sind, im Zustandsregister gespeichert wird, um die Art und Weise darzustellen, wie der Rest korrigiert werden muß.

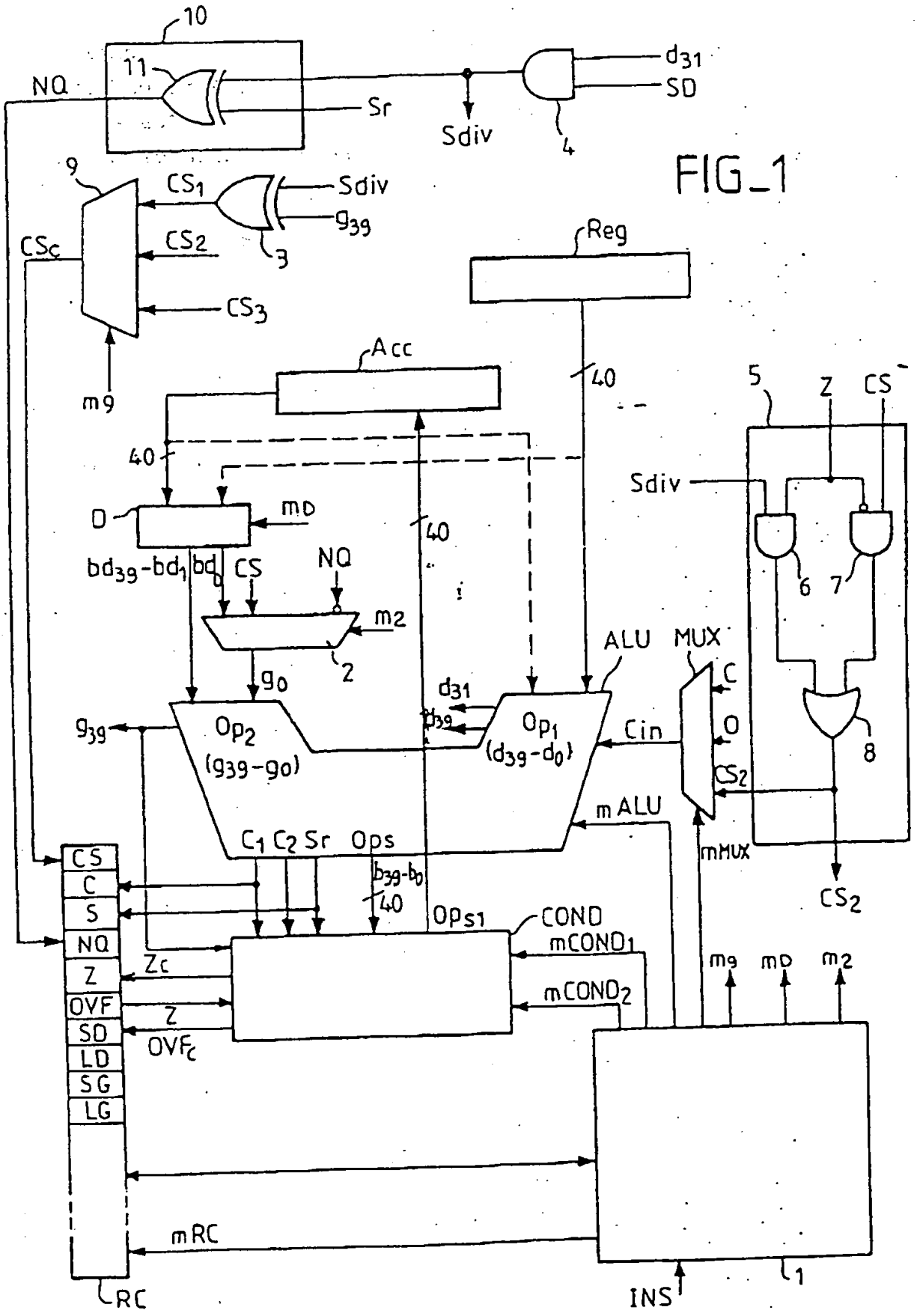
10
11
NQ
Sr
Sdiv
d31
SD
4
9
CS1
Sdiv
g39
CS2
3
CSc
CS3
m9
Reg
Acc
40
5
Z
CS
Sdiv
40
D
mD
bd39-bd1
bd0
CS
NQ
m2
2
6
7
40
ALU
MUX
g0
d31
b39
g39
Op2
(g39-g0)
Op1
(d39-d0)
Cin
C
0
CS2
8
mALU
C1 C2 Sr Ops
mMUX
CS2
CS
C
S
NQ
Z
OVF
SD
LD
SG
LG
b39-b0
40
Ops1
COND
mCOND1
mCOND2
Zc
Z
OVFc
m9
mD
m2
mRC
RC
INS
1

FIG_1

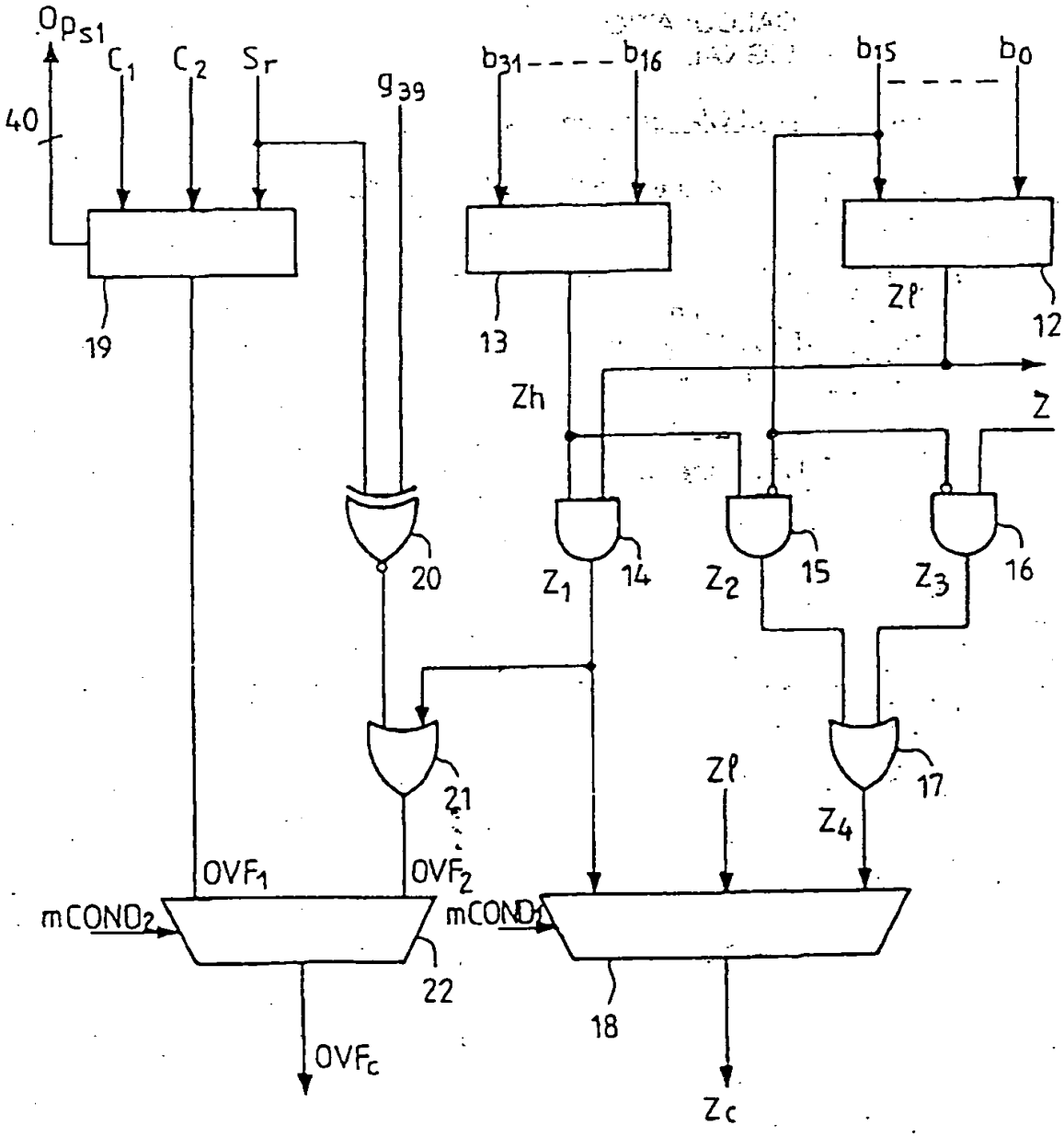
Op_S1
C1
C2
Sr
g39
b31 - - - b16
b15
b0
40
19
13
Zl
12
Zh
Z
20
Z1
14
Z2
15
Z3
16
21
Zl
17
Z4
OVF1
OVF2
mCOND2
mCOND1
22
18
OVFc
Zc

FIG_2